FAST DISTRIBUTED COORDINATE DESCENT FOR NON-STRONGLY CONVEX LOSSES*

Olivier Fercoq Zheng Qu Peter Richtárik Martin Takáč

School of Mathematics, University of Edinburgh, Edinburgh, EH9 3JZ, United Kingdom

ABSTRACT

We propose an efficient distributed randomized coordinate descent method for minimizing regularized non-strongly convex loss functions. The method attains the optimal $O(1/k^2)$ convergence rate, where k is the iteration counter. The core of the work is the theoretical study of stepsize parameters. We have implemented the method on Archer—the largest supercomputer in the UK—and show that the method is capable of solving a (synthetic) LASSO optimization problem with 50 billion variables.

Index Terms— Coordinate descent, distributed algorithms, acceleration.

1. INTRODUCTION

In this paper we are concerned with solving regularized convex optimization problems in *huge dimensions* in cases when the loss being minimized is *not* strongly convex. That is, we consider the problem

$$\min_{x\in\mathbb{R}^d} L(x) := f(x) + R(x), \tag{1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a convex differentiable loss function, d is huge, and $R(x) \equiv \sum_{i=1}^d R_i(x^i)$, where $R_i : \mathbb{R} \to \mathbb{R}\cup\{+\infty\}$ are (possibly nonsmooth) convex regularizers and x^i denotes the i-th coordinate of x. We make the following technical assumption about f: there exists a n-by-d matrix $\mathbf{A}$ such that for all $x, h \in \mathbb{R}^d$,

$$f(x+h) \leqslant f(x) + (f'(x))^\top h + \tfrac{1}{2}h^\top \mathbf{A}^\top \mathbf{A} h. \tag{2}$$

For examples of regularizers R and loss functions f satisfying the above assumptions, relevant to machine learning applications, we refer the reader to [1, 2, 3].

It is increasingly the case in modern applications that the data describing the problem (encoded in $\mathbf{A}$ and R in the above model) is so large that it does not fit into the RAM of a single computer. In such a case, unless the application at hand can tolerate slow performance due to frequent HDD reads/writes, it is often necessary to distribute the data among the nodes of a cluster and solve the problem in a distributed manner.

While efficient distributed methods exist in cases when the regularized loss L is strongly convex (e.g., Hydra [3]), here we assume that L is *not* strongly convex. Problems of this type arise frequently: for instance, in the SVM dual, f is typically a non-strongly convex quadratic, d is the number of examples, n is the number of features, $\mathbf{A}$ encodes the data, and R is a 0-∞ indicator function encoding box constraints (e.g., see [2]). If $d > n$, then L will not be strongly convex.

In this paper we propose "Hydra^2" (Hydra "squared"; Algorithm 1) – the first distributed stochastic coordinate descent (CD) method with fast $O(1/k^2)$ convergence rate for our problem. The method can be seen both as a specialization of the APPROX algorithm [4] to the distributed setting, or as an accelerated version of the Hydra algorithm [3] (Hydra converges as $O(1/k)$ for our problem). The core of the paper forms the development of *new stepsizes*, and new efficiently computable bounds on the stepsizes proposed for distributed CD methods in [3]. We show that Hydra^2 is able to solve a big data problem with d equal to *50 billion.*

2. THE ALGORITHM

Assume we have c nodes/computers available. In Hydra^2 (Algorithm 1), the coordinates $i \in [d] := \{1, 2, \ldots, d\}$ are first partitioned into c sets $\{\mathcal{P}_l,\ l = 1, \ldots, c\}$, each of size $|P_l| = s := d/c$. The columns of $\mathbf{A}$ are partitioned accordingly, with those belonging to partition $\mathcal{P}_l$ stored on node l. During one iteration, all computers $l \in \{1, \ldots, c\}$, in parallel, pick a subset $\hat{S}_l$ of τ coordinates from those they own, i.e., from $\mathcal{P}_l$, uniformly at random, where $1 \leqslant \tau \leqslant s$ is a parameter of the method (Step 6). From now on we denote by $\hat{S}$ the union of all these sets, $\hat{S} := \cup_l \hat{S}_l$, and refer to it by the name *distributed sampling*.

Hydra^2 maintains two sequences of iterates: $u_k, z_k \in \mathbb{R}^d$. Note that this is usually the case with accelerated/fast gradient-type algorithms [5, 6, 7]. Also note that the output of the method is a linear combination of these two vectors. These iterates are stored and updated in a distributed way, with the i-th coordinate stored on computer l if $i \in \mathcal{P}_l$. Once computer l picks $\hat{S}_l$, it computes for each $i \in \hat{S}_l$ an update scalar t_k^i, which is then used to update z_k^i and u_k^i, in parallel (using the multicore capability of computer l).

The main work is done in Step 8 where the update scalars

*THANKS TO EPSRC FOR FUNDING VIA GRANTS EP/K02325X/1, EP/I017127/1 AND EP/G036136/1.

Algorithm 1 Hydra^2

INPUT: $\{\mathcal{P}_l\}_{l=1}^c$, $1 \leqslant \tau \leqslant s$, $\{\mathbf{D}_{ii}\}_{i=1}^d$, $z_0 \in \mathbb{R}^d$
set $\theta_0 = \tau/s$ and $u_0 = 0$
for $k \geqslant 0$ **do**
  $z_{k+1} \leftarrow z_k$, $u_{k+1} \leftarrow u_k$
  for each computer $l \in \{1, \dots, c\}$ **in parallel do**
    pick a random set of coordinates $\hat{S}_l \subseteq \mathcal{P}_l$, $|\hat{S}_l| = \tau$
    for each $i \in \hat{S}_l$ **in parallel do**
      $t_k^i = \operatorname{argmin}_t f_i'(\theta_k^2 u_k + z_k)t + \frac{s\theta_k \mathbf{D}_{ii}}{2\tau} t^2 + R_i(z_k^i + t)$
      $z_{k+1}^i \leftarrow z_k^i + t_k^i$, $u_{k+1}^i \leftarrow u_k^i - (\frac{1}{\theta_k^2} - \frac{s}{\tau\theta_k}) t_k^i$
    end parallel for
  end parallel for
  $\theta_{k+1} = \frac{1}{2}(\sqrt{\theta_k^4 + 4\theta_k^2} - \theta_k^2)$
end for
OUTPUT: $\theta_k^2 u_{k+1} + z_{k+1}$

t_k^i are computed. The partial derivatives are computed at $(\theta_k^2 u_k + z_k)$. For the algorithm to be efficient, the computation should be performed *without computing the sum* $\theta_k^2 u_k + z_k$ (see [4] for details on how this can be done).

Steps 8 and 9 depend on a deterministic scalar sequence θ_k, which is being updated in Step 12 as in [6]. Note that by taking $\theta_k = \theta_0$ for all k, u_k remains equal to 0, and Hydra^2 reduces to Hydra [3].

The output of the algorithm is $x_{k+1} = (\theta_k^2 u_{k+1} + z_{k+1})$. We only need to compute this vector sum at the end of the execution and when we want to track $L(x_k)$. Note that one should not evaluate x_k and $L(x_k)$ at each iteration since these computations have a non-negligible cost.

3. CONVERGENCE RATE

The magnitude of $\mathbf{D}_{ii} > 0$ directly influences the size of the update t_k^i. In particular, note that when there is no regularizer ($R_i = 0$), then $t_k^i = 2\tau f_i'(\theta_k^2 u_k + z_k)(s\theta_k \mathbf{D}_{ii})^{-1}$. That is, small $\mathbf{D}_{ii}$ leads to a larger "step" t_k^i which is used to update z_k^i and u_k^i. For this reason we refer to $\{\mathbf{D}_{ii}\}_{i=1}^d$ as *stepsize parameters*. Naturally, some technical assumptions on $\{\mathbf{D}_{ii}\}_{i=1}^d$ should be made in order to guarantee the convergence of the algorithm. The so-called ESO (Expected Separable Overapproximation) assumption has been introduced in this scope. For $h \in \mathbb{R}^d$, denote $h^{\hat{S}} := \sum_{i \in \hat{S}} h^i e_i$, where e_i is the ith unit coordinate vector.

Assumption 3.1 (ESO). *Assume that for all $x \in \mathbb{R}^d$ and $h \in \mathbb{R}^d$ we have*

$$\mathbb{E}\left[f(x + h^{\hat{S}})\right] \leqslant f(x) + \frac{\mathbb{E}[|\hat{S}|]}{d}\left((f'(x))^\top h + \frac{1}{2} h^\top \mathbf{D} h\right), \quad (3)$$

where $\mathbf{D}$ is a diagonal matrix with diag. elements $\mathbf{D}_{ii} > 0$ and $\hat{S}$ is the distributed sampling described above.

The above ESO assumption involves the smooth function f, the sampling $\hat{S}$ and the parameters $\{\mathbf{D}_{ii}\}_{i=1}^d$. It has been first introduced by Richtárik and Takáč [2] for proposing a generic approach in the convergence analysis of the Parallel Coordinate Descent Methods (PCDM). Their generic approach boils down the convergence analysis of the whole class of PCDMs to the problem of finding proper parameters which make the ESO assumption hold. The same idea has been extended to the analysis of many variants of PCDM, including the Accelerated Coordinate Descent algorithm [4] (APPROX) and the Distributed Coordinate Descent method [3] (Hydra).

In particular, the following complexity result, under the ESO assumption 3.1, can be deduced from [4, Theorem 3] using Markov inequality:

Theorem 3.2 ([4]). *Let $x_* \in \mathbb{R}^d$ be any optimal point of* (1)*,*

$$C_1 := \left(1 - \tfrac{\tau}{s}\right)(L(x_0) - L(x_*)),$$

$$C_2 := \tfrac{1}{2}(x_0 - x_*)^\top \mathbf{D}(x_0 - x_*),$$

and choose $0 < \rho < 1$, $\epsilon < L(x_0) - L(x_*)$ and

$$k \geqslant \frac{2s}{\tau}\left(\sqrt{\frac{C_1 + C_2}{\rho\epsilon}} - 1\right) + 1. \quad (4)$$

Then under Assumption 3.1, the iterates $\{x_k\}_{k \geqslant 1}$ of Algorithm 1 satisfy $\mathbf{Prob}(L(x_k) - L(x_*) \leqslant \epsilon) \geqslant 1 - \rho$.

The bound on the number of iterations k in (4) suggests to choose the smallest possible $\{\mathbf{D}_{ii}\}_{i=1}^d$ satisfying the ESO assumption 3.1.

4. STEPSIZES

In this section we will study stepsize parameters $\mathbf{D}_{ii}$ for which the ESO assumption is satisfied.

4.1. New stepsizes

For any matrix $\mathbf{G} \in \mathbb{R}^{d \times d}$, denote by $D^{\mathbf{G}}$ the diagonal matrix such that $D_{ii}^{\mathbf{G}} = \mathbf{G}_{ii}$ for all i and $D_{ij}^{\mathbf{G}} = 0$ for $i \neq j$. We denote by $B^{\mathbf{G}} \in \mathbb{R}^{d \times d}$ the block matrix associated with the partition $\{\mathcal{P}_1, \dots, \mathcal{P}_c\}$ such that $B_{ij}^{\mathbf{G}} = \mathbf{G}_{ij}$ whenever $\{i, j\} \subset \mathcal{P}_l$ for some l, and $B_{ij}^{\mathbf{G}} = 0$ otherwise.

Let ω_j be the number of nonzeros in the jth row of $\mathbf{A}$ and ω_j' be the number of "partitions active at row j", i.e., the number of indexes $l \in \{1, \dots, c\}$ for which the set $\{i \in \mathcal{P}_l : \mathbf{A}_{ji} \neq 0\}$ is nonempty. Note that since $\mathbf{A}$ does not have an empty row or column, we know that $1 \leqslant \omega_j \leqslant n$ and $1 \leqslant \omega_j' \leqslant c$. Moreover, we have the following characterization of these two quantities. For notational convenience we denote $\mathbf{M}_j = \mathbf{A}_{j:}^\top \mathbf{A}_{j:}$ for all $j \in \{1, \dots, n\}$.

Lemma 4.1. *For $j \in \{1,\dots,n\}$ we have:*

$$\omega_j = \max\{x^\top \mathbf{M}_j x \ : \ x^\top D^{\mathbf{M}_j} x \leqslant 1\}, \tag{5}$$
$$\omega_j' = \max\{x^\top \mathbf{M}_j x \ : \ x^\top B^{\mathbf{M}_j} x \leqslant 1\}. \tag{6}$$

Proof. For $l \in \{1,\dots,c\}$ and $y \in \mathbb{R}^d$, denote $y^{(l)} := (y^i)_{i\in\mathcal{P}_l}$. That is, $y^{(l)}$ is the subvector of y composed of coordinates belonging to partition $\mathcal{P}_l$. Then we have:

$$x^\top \mathbf{M}_j x = (\sum_{l=1}^{c} \mathbf{A}_{j:}^{(l)} x^{(l)})^2, \ \ x^\top B^{\mathbf{M}_j} x = \sum_{l=1}^{c} (\mathbf{A}_{j:}^{(l)} x^{(l)})^2.$$

Let $S' = \{l \ : \ \mathbf{A}_{j:}^{(l)} \neq 0\}$, then $\omega_j' = |S'|$ and by Cauchy-Schwarz we have

$$(\sum_{l\in S'} \mathbf{A}_{j:}^{(l)} x^{(l)})^2 \ \ \leqslant \ \ \omega_j' \sum_{l\in S'} (\mathbf{A}_{j:}^{(l)} x^{(l)})^2.$$

Equality is reached when $\mathbf{A}_{j:}^{(l)} x^{(l)} = \alpha$ for some constant α for all $l \in S'$ (this is feasible since the subsets $\{\mathcal{P}_1,\dots,\mathcal{P}_c\}$ are disjoint). Hence, we proved (6). The characterization (5) follows from (6) by setting $c = d$. □

We shall also need the following lemma.

Lemma 4.2 ([3])**.** *Fix arbitrary $\mathbf{G} \in \mathbb{R}^{d\times d}$ and $x \in \mathbb{R}^d$ and let $s_1 = \max\{1, s-1\}$. Then $\mathbb{E}[(x^{\hat{S}})^\top \mathbf{G} x^{\hat{S}}]$ is equal to*

$$\tfrac{\tau}{s}\left[\alpha_1 x^\top D^{\mathbf{G}} x + \alpha_2 x^\top \mathbf{G} x + \alpha_3 x^\top (\mathbf{G} - B^{\mathbf{G}}) x\right], \tag{7}$$

where $\alpha_1 = 1 - \frac{\tau-1}{s_1}$, $\alpha_2 = \frac{\tau-1}{s_1}$, $\alpha_3 = \frac{\tau}{s} - \frac{\tau-1}{s_1}$.

Below is the main result of this section.

Theorem 4.1. *For a convex differentiable function f satisfying* (2) *and a distributed sampling $\hat{S}$ described in Section 2, the ESO assumption 3.1 is satisfied for*

$$\mathbf{D}_{ii}^1 \equiv \mathbf{D}_{ii} = \sum_{j=1}^{n} \alpha_j^* \mathbf{A}_{ji}^2, \quad i = 1,\dots,d \ , \tag{8}$$

where

$$\begin{aligned} &\alpha_j^* := \alpha_{1,j}^* + \alpha_{2,j}^*, \ \ \alpha_{1,j}^* = 1 + \tfrac{(\tau-1)(\omega_j-1)}{s_1}, \\ &\alpha_{2,j}^* = (\tfrac{\tau}{s} - \tfrac{\tau-1}{s_1}) \tfrac{\omega_j'-1}{\omega_j'} \omega_j. \end{aligned} \tag{9}$$

Proof. Fix any $h \in \mathbb{R}^d$. It is easy to see that $\mathbb{E}[(f'(x))^\top h^{\hat{S}}] = \frac{\mathbb{E}[|\hat{S}|]}{d}(f'(x))^\top h)$; this follows by noting that $\hat{S}$ is a uniform sampling (we refer the reader to [2] and [8] for more identities of this type). In view of (2), we only need to show that

$$\mathbb{E}\left[(h^{\hat{S}})^\top \mathbf{A}^\top \mathbf{A} h^{\hat{S}}\right] \leqslant \tfrac{\mathbb{E}[|\hat{S}|]}{d} h^\top \mathbf{D} h \ . \tag{10}$$

By applying Lemma 4.2 with $\mathbf{G} = \mathbf{M}_j$ and using Lemma 4.1, we get:

$$\begin{aligned} &\mathbb{E}\left[(h^{\hat{S}})^\top \mathbf{M}_j h^{\hat{S}}\right] \\ &\leqslant \tfrac{\tau}{s}[\alpha_1 h^\top D^{\mathbf{M}_j} h + \alpha_2 \omega_j h^\top D^{\mathbf{M}_j} h + \alpha_3 (1 - \tfrac{1}{\omega_j'}) h^\top \mathbf{M}_j h] \\ &\leqslant \tfrac{\tau}{s}[\alpha_1 + \alpha_2 \omega_j + \alpha_3 (1 - \tfrac{1}{\omega_j'}) \omega_j] h^\top D^{\mathbf{M}_j} h \\ &= \tfrac{\mathbb{E}[|\hat{S}|]}{d} \alpha_j^* h^\top D^{\mathbf{M}_j} h. \end{aligned} \tag{11}$$

Since $\mathbb{E}[(h^{\hat{S}})^\top \mathbf{A}^\top \mathbf{A} h^{\hat{S}}] = \sum_{j=1}^n \mathbb{E}[(h^{\hat{S}})^\top \mathbf{M}_j h^{\hat{S}}]$, (10) can be obtained by summing up (11) over j from 1 to n. □

4.2. Existing stepsizes

For simplicity, let $\mathbf{M} := \mathbf{A}^\top \mathbf{A}$. Define:

$$\begin{aligned} \sigma &:= \max\{x^\top \mathbf{M} x : x \in \mathbb{R}^d; x^\top D^{\mathbf{M}} x \leqslant 1\}, \\ \sigma' &:= \max\{x^\top \mathbf{M} x : x \in \mathbb{R}^d; x^\top B^{\mathbf{M}} x \leqslant 1\}. \end{aligned} \tag{12}$$

The quantities σ and σ' are identical to those defined in [3] (although the definitions are slightly different). The following stepsize parameters have been introduced in [3]:

Lemma 4.3 ([3])**.** *The ESO assumption 3.1 is satisfied for*

$$\mathbf{D}_{ii}^2 \equiv \mathbf{D}_{ii} = \beta^* \sum_{j=1}^{n} \mathbf{A}_{ji}^2, \ \ i = 1,\dots,d \ , \tag{13}$$

where

$$\begin{aligned} &\beta^* := \beta_1^* + \beta_2^* \\ &\beta_1^* = 1 + \tfrac{(\tau-1)(\sigma-1)}{s_1}, \ \ \beta_2^* = \left(\tfrac{\tau}{s} - \tfrac{\tau-1}{s_1}\right) \tfrac{\sigma'-1}{\sigma'} \sigma. \end{aligned} \tag{14}$$

4.3. Known bounds on existing stepsizes

In general, the computation of σ and σ' can be done using the Power iteration method [9]. However, the number of operations required in each iteration of the Power method is at least twice as the number of nonzero elements in $\mathbf{A}$. Hence the only computation of the parameters σ and σ' would already require quite a few number of passes through the data. Instead, if one could provide some easily computable upper bound for σ and σ', where by 'easily computable' we mean computable by only one pass through the data, then we can run Algorithm 1 immediately without spending too much time on the computation of σ and σ'. Note that the ESO assumption will still hold when σ and σ' in (13) are replaced by some upper bounds.

In [3], the authors established the following bound

$$\beta^* \leqslant 2\beta_1^* = 2(1 + \tfrac{(\tau-1)(\sigma-1)}{s_1}) \ , \tag{15}$$

which is independent of the partition $\{\mathcal{P}_l\}_{l=1,\dots,c}$. This bound holds for $\tau \geqslant 2$. Further, they showed that

$$\sigma \leqslant \max_j \omega_j \ . \tag{16}$$

Then in view of (15), (16) and Lemma 4.3, the ESO assumption 3.1 is also satisfied for the following easily computable parameters:

$$\mathbf{D}^3_{ii} \equiv \mathbf{D}_{ii} = 2\left(1 + \tfrac{\tau-1}{s_1}(\max_j \omega_j - 1)\right)\sum_{j=1}^n \mathbf{A}^2_{ji}. \quad (17)$$

4.4. Improved bounds on existing stepsizes

In what follows, we show that both (15) and (16) can be improved so that smaller parameters $\{\mathbf{D}_{ii}\}_{i=1}^d$ are allowed in the algorithm.

Lemma 4.4. *Suppose $\tau \geqslant 2$ (note that then $s \geqslant \tau \geqslant 2$). For all $1 \leqslant \eta \leqslant s$ the following holds*

$$\left(\tfrac{\tau}{s} - \tfrac{\tau-1}{s-1}\right)\eta \leqslant \tfrac{1}{\tau-1}\left(1 + \tfrac{(\tau-1)(\eta-1)}{s-1}\right) \quad (18)$$

Proof. Both sides of the inequality are linear functions of η. It therefore suffices to verify that the inequality holds for $\eta = 1$ and $\eta = s$, which can be done. □

The following result is an improvement on (15), which was shown in [3, Lemma 2].

Lemma 4.5. *If $\tau \geqslant 2$, then $\beta^* \leqslant (1 + \frac{1}{\tau-1})\beta_1^*$.*

Proof. We only need to apply Lemma 4.4 to (14) with $\eta = \sigma$, and additionally use the bound $\frac{\sigma'-1}{\sigma'} \leqslant 1$. This gives $\beta_2^* \leqslant \beta_1^*/(\tau-1)$, from which the result follows. □

Remark 4.1. *Applying the same reasoning to the new stepsizes α_j^* we can show that $\alpha_j^* \leqslant (1+1/(\tau-1))\alpha_{1,j}^*$ for all $j \in \{1,\dots,n\}$. Remark that this is not as useful as Lemma 4.5 since $\alpha_{2,j}^*$ does not need to be approximated (ω_j' is easily computable). However, the same conclusion holds for the new stepsize parameters: whatever the partition $\{\mathcal{P}_l\}_{l=1,\dots,c}$ is, each parameter $\mathbf{D}_{ii}$ is at most $1 + 1/(\tau-1)$ times than the smallest one we can use by choosing an optimal partition, if the latter one exists.*

We next give a better upper bound of σ than (16).

Lemma 4.6. *If we let*

$$v_i := \frac{\sum_{j=1}^n \omega_j \mathbf{A}^2_{ji}}{\sum_{j=1}^n \mathbf{A}^2_{ji}}, \;\; i = 1,\dots,d, \quad (19)$$

then $\sigma \leqslant \tilde{\sigma} := \max_i v_i$.

Proof. In view of Lemma 4.1, we know that

$$\mathbf{M} = \sum_{j=1}^n \mathbf{M}_j \preccurlyeq \sum_{j=1}^n \omega_j D^{\mathbf{M}_j} = \mathrm{Diag}(v) D^{\mathbf{M}} \preccurlyeq \max_i v_i D^{\mathbf{M}}.$$

The rest follows from the definition of σ. □

By combining Lemma 4.3, Lemma 4.5 and Lemma 4.6, we obtain the following smaller (compared to (17)) admissible and easily computable stepsize parameters:

$$\mathbf{D}^4_{ii} \equiv \mathbf{D}_{ii} = \tfrac{\tau}{\tau-1}\left(1 + \tfrac{(\tilde{\sigma}-1)(\tau-1)}{s-1}\right)\sum_{j=1}^n \mathbf{A}^2_{ji}. \quad (20)$$

4.5. Comparison between the new and old stepsizes

So far we have seen four different admissible stepsize parameters. For ease of reference, let us call $\{\mathbf{D}^1_{ii}\}_{i=1}^d$ the new parameters defined in (8), $\{\mathbf{D}^2_{ii}\}_{i=1}^d$ the existing one given in [3] (see (13)), $\{\mathbf{D}^3_{ii}\}_{i=1}^d$ the upper bound of $\{\mathbf{D}^2_{ii}\}_{i=1}^d$ used in [3] (see (17)) and $\{\mathbf{D}^4_{ii}\}_{i=1}^d$ the new upper bound of $\{\mathbf{D}^2_{ii}\}_{i=1}^d$ defined in (20).

The next lemma compares the four admissible ESO parameters. The lemma implies that $\mathbf{D}^1$ and $\mathbf{D}^2$ are uniformly smaller (i.e., better – see Theorem 3.2) than $\mathbf{D}^4$ and $\mathbf{D}^3$. However, $\mathbf{D}^2$ involves quantities which are hard to compute (e.g., σ). Moreover, $\mathbf{D}^4$ is always better than $\mathbf{D}^3$.

Lemma 4.7. *Let $\tau \geqslant 2$. The following holds for all i:*

$$\mathbf{D}^1_{ii} \leqslant \mathbf{D}^4_{ii} \leqslant \mathbf{D}^3_{ii} \quad (21)$$

$$\mathbf{D}^2_{ii} \leqslant \mathbf{D}^4_{ii} \leqslant \mathbf{D}^3_{ii} \quad (22)$$

Proof. We only need to show that $\mathbf{D}^1_{ii} \leqslant \mathbf{D}^4_{ii}$, the other relations are already proved. It follows from (19) that for all i we have

$$\textstyle\sum_{j=1}^n (\omega_j - \tilde{\sigma})\mathbf{A}^2_{ji} \leqslant 0 \;\; . \quad (23)$$

Moreover, letting

$$\tilde{\sigma}' = \max_j \omega_j' \quad (24)$$

and using Lemma 4.4 with $\eta = \tilde{\sigma}$, we get

$$(\tfrac{\tau}{s} - \tfrac{\tau-1}{s-1})\tfrac{\tilde{\sigma}'-1}{\tilde{\sigma}'}\tilde{\sigma} \leqslant \tfrac{1}{\tau-1}(1 + \tfrac{(\tilde{\sigma}-1)(\tau-1)}{s-1}). \quad (25)$$

Now, for all i, we can write

$$\begin{aligned}
&\mathbf{D}^1_{ii} - \mathbf{D}^4_{ii} \\
&\overset{(8)+(9)+(20)}{=} \sum_{j=1}^n \mathbf{A}^2_{ji}\left(\tfrac{\tau-1}{s-1}(\omega_j - \tilde{\sigma}) + \right. \\
&\qquad \left.(\tfrac{\tau}{s} - \tfrac{\tau-1}{s-1})\tfrac{\omega_j'-1}{\omega_j'}\omega_j - \tfrac{1}{\tau-1}(1 + \tfrac{(\tilde{\sigma}-1)(\tau-1)}{s-1})\right) \\
&\overset{(23)}{\leqslant} \sum_{j=1}^n \left((\tfrac{\tau}{s} - \tfrac{\tau-1}{s-1})\tfrac{\omega_j'-1}{\omega_j'}\omega_j - \tfrac{1}{\tau-1}(1 + \tfrac{(\tilde{\sigma}-1)(\tau-1)}{s-1})\right)\mathbf{A}^2_{ji} \\
&\overset{(24)+(25)}{\leqslant} (\tfrac{\tau}{s} - \tfrac{\tau-1}{s_1})\tfrac{\tilde{\sigma}'-1}{\tilde{\sigma}'}\sum_{j=1}^n (\omega_j - \tilde{\sigma})\mathbf{A}^2_{ji} \overset{(23)}{\leqslant} 0 \;\; .
\end{aligned}$$

□

Note that we do not have a simple relation between $\{\mathbf{D}^1_{ii}\}_{i=1}^d$ and $\{\mathbf{D}^2_{ii}\}_{i=1}^d$. Indeed, for some coordinates i the new stepsize parameters $\{\mathbf{D}^1_{ii}\}_{i=1}^d$ could be smaller than $\{\mathbf{D}^2_{ii}\}_{i=1}^d$, see Figure 1. in the next section for an illustration.

5. NUMERICAL EXPERIMENTS

In this section we will show some preliminary computational result and demonstrate that Hydra^2 can, indeed, outperform (non-accelerated) Hydra. We also show that the runtime of Hydra^2 is (as expected) less than twice more expensive when compared with Hydra, but the convergence speed is significantly better on non-strongly convex problems.

The experiments were performed on two problems:

1. **Dual of SVM:** This problem can be formulated as finding $x \in [0,1]^d$ that minimizes

$$L(x) = \frac{1}{2\lambda d^2}\sum_{j=1}^{n}(\sum_{i=1}^{d} b^i \mathbf{A}_{ji}x^i)^2 - \frac{1}{d}\sum_{i=1}^{d} x^i + I_{[0,1]^d}(x),$$

Here d is the number of examples, each row of the matrix $\mathbf{A}$ corresponds to a feature, $b \in \mathbb{R}^d$, $\lambda > 0$ and $I_{[0,1]^d}$ denotes the indicator function of the set $[0,1]^d$, defined as

$$I_{[0,1]^d}(x) = \begin{cases} 0 & \text{if } x \in [0,1]^d, \\ +\infty & \text{if } x \notin [0,1]^d. \end{cases}$$

2. **LASSO:**

$$L(x) = \frac{1}{2}\|\mathbf{A}x - b\|^2 + \lambda\|x\|_1. \qquad (26)$$

The computation were performed on Archer (`http://archer.ac.uk/`), which is UK #1 supercomputer .

5.1. Benefit from the new stepsizes

In this section we compare the four different stepsize parameters and show how they influence the convergence of Hydra^2 through numerical experiments. The dataset used in this section is *astro-ph* dataset.[1] The problem that we solve is the SVM dual problem, see [10, 11].

Figure 1 displays the values of $\mathbf{D}^1_{ii}$, $\mathbf{D}^2_{ii}$, $\mathbf{D}^3_{ii}$ and $\mathbf{D}^4_{ii}$ with respect to different i in $\{1,\dots,d\}$ for $(c,\tau) = (32,10)$ on the *astro-ph* dataset. For this particular normalized dataset (the diagonal elements of $\mathbf{A}^\top\mathbf{A}$ are all equal to 1), the parameters $\{\mathbf{D}^2_{ii}\}_i$ are equal to a constant for all i, the same for $\{\mathbf{D}^3_{ii}\}_i$ and $\{\mathbf{D}^4_{ii}\}_i$. Recall that according to Theorem 3.2, a better convergence rate is guaranteed if smaller stepsize parameters $\{\mathbf{D}_{ii}\}_i$ are used. Hence it is clear that $\{\mathbf{D}^1_{ii}\}_i$ and $\{\mathbf{D}^2_{ii}\}_i$ are better choices than $\{\mathbf{D}^3_{ii}\}_i$ and $\{\mathbf{D}^4_{ii}\}_i$. However, as we mentioned, computing the parameters $\{\mathbf{D}^1_{ii}\}_i$ would require a considerable computational effort, while computing $\{\mathbf{D}^2_{ii}\}_i$ is as easy as passing one time the dataset. For this relatively small astro-ph dataset, the time is about 1 minute for computing $\{\mathbf{D}^2_{ii}\}_i$ and less than 1 second for $\{\mathbf{D}^1_{ii}\}_i$.

[1] Astro-ph is a binary classification problem which consist of abstracts of papers from physics. The dataset consist of $d = 29,882$ samples and the feature space has dimension $n = 99,757$.

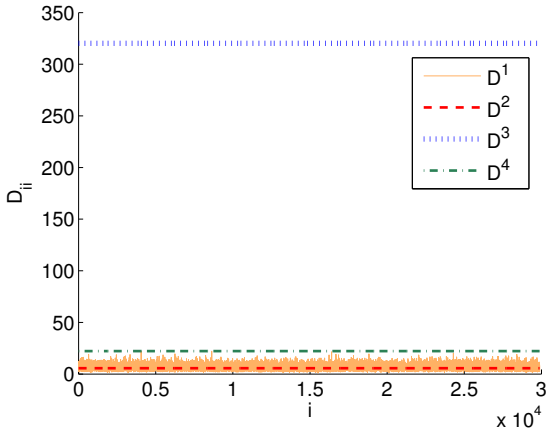

Fig. 1: Plots of i v.s. $\mathbf{D}^1_{ii}$, i v.s. $\mathbf{D}^2_{ii}$, i v.s. $\mathbf{D}^3_{ii}$ and i v.s. $\mathbf{D}^4_{ii}$

In order to investigate the benefit of the new stepsize parameters, we solved the SVM dual problem on the *astro-ph* dataset for $(c,\tau) = (32,10)$, using different stepsize parameters. Figure 2 shows evolutions of the duality gap, obtained by using the four stepsize parameters mentioned previously. We see clearly from the figure that smaller stepsize parameters yield better convergence speed, as what is predicted by Theorem 3.2. Moreover, using our easily computable new stepsize parameters $\{\mathbf{D}^1_{ii}\}_i$, we achieve comparable convergence speed with respect to the existing ESO parameter $\{\mathbf{D}^2_{ii}\}_i$.

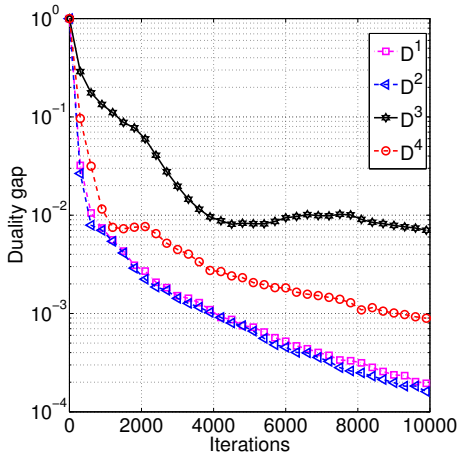

Fig. 2: Duality gap v.s. number of iterations, for 4 different stepsize parameters

5.2. Hydra^2 vs Hydra

In this section, we report experimental results comparing Hydra with Hydra^2 on the LASSO problem. We generated a sparse matrix $\mathbf{A}$ having the same block angular structure as the one used in [3, Sec 7.], with $d = 50$ billion, $c = 8$, $s = 62,500,000$ (note that $d = cs$) and $n = 500,000$.

The average number of nonzero elements per row of $\mathbf{A}$, i.e., $\sum_j \omega_j/n$ is $5,000$, and the maximal number of nonzero elements in a row, i.e., $\max_j \omega_j$ is $201,389$. The dataset size is 400GB. We show in Figure 3 the error decrease with respect to the number of iterations, plotted in log scale. It is clear that Hydra^2 provides better iteration convergence rate than Hydra. Moreover, we see from Figure 4 that the speedup in terms of the number of iterations is sufficiently large so that Hydra^2 converges faster even in time than Hydra even though the run time of Hydra^2 per iteration is more expensive than Hydra.

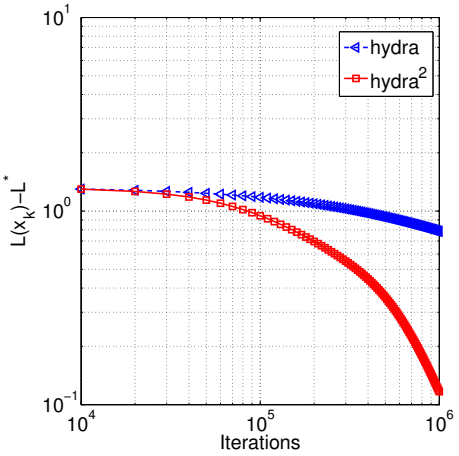

Fig. 3: Evolution of $L(x_k) - L^*$ in number of iterations, plotted in log scale.

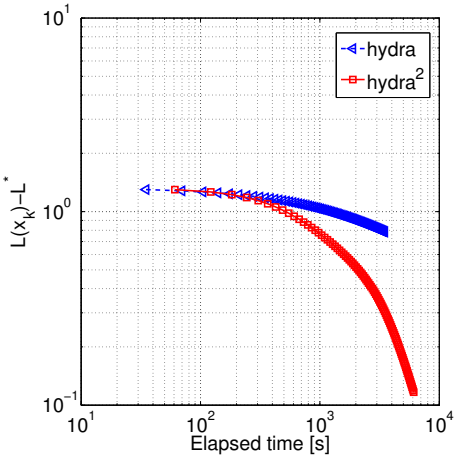

Fig. 4: Evolution of $L(x_k) - L^*$ in time, plotted in log scale.

6. REFERENCES

[1] Joseph K. Bradley, Aapo Kyrola, Danny Bickson, and Carlos Guestrin, "Parallel coordinate descent for l1-regularized loss minimization," in *ICML 2011*, 2011.

[2] Peter Richtárik and Martin Takáč, "Parallel coordinate descent methods for big data optimization," *arXiv:1212.0873*, 2012.

[3] Peter Richtárik and Martin Takáč, "Distributed coordinate descent method for learning with big data," *arXiv:1310.2059*, 2013.

[4] Olivier Fercoq and Peter Richtárik, "Accelerated, parallel and proximal coordinate descent," *arXiv:1312.5799*, 2013.

[5] Yurii Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.

[6] Paul Tseng, "On accelerated proximal gradient methods for convex-concave optimization," *Submitted to SIAM Journal on Optimization*, 2008.

[7] Amir Beck and Marc Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[8] Olivier Fercoq and Peter Richtárik, "Smooth minimization of nonsmooth functions with parallel coordinate descent methods," *arXiv:1309.5885*, 2013.

[9] Grégoire Allaire and Sidi Mahmoud Kaber, *Numerical linear algebra*, vol. 55 of *Texts in Applied Mathematics*, Springer, New York, 2008, Translated from the 2002 French original by Karim Trabelsi.

[10] Shai Shalev-Shwartz and Tong Zhang, "Stochastic dual coordinate ascent methods for regularized loss," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 567–599, 2013.

[11] Martin Takáč, Avleen Bijral, Peter Richtárik, and Nathan Srebro, "Mini-batch primal and dual methods for SVMs," in *ICML 2013*, 2013.